HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80528-9599

08-02-00

PATENT APPLICATION A

ATTORNEY DOCKET NO. 10003696-1

jc872 U.S. PTO 07/31/00

IN THE U.S. PATENT AND TRADEMARK OFFICE
Patent Application Transmittal Letter

ASSISTANT COMMISSIONER FOR PATENTS
Washington, D.C. 20231

Sir:

jc862 U.S. PTO 09/628919 07/31/00

Transmitted herewith for filing under 37 CFR 1.53(b) is a(n): (X) Utility ( ) Design
(X) original patent application,
( ) continuation-in-part application

INVENTOR(S): **Jeffrey B. THOMPSON**

TITLE: **METHOD AND SYSTEM FOR EXTENDABLE CLASS-BASED SHARED DATA-TYPES**

Enclosed are:

(X) The Declaration and Power of Attorney. ( ) signed (X) unsigned or partially signed
(X) __2__ sheets of drawings (one set) ( ) Associate Power of Attorney
( ) Form PTO-1449 ( ) Information Disclosure Statement and Form PTO-1449
( ) Priority document(s) ( )(Other)_____ (fee $_____)

CLAIMS AS FILED BY OTHER THAN A SMALL ENTITY				
(1) FOR	(2) NUMBER FILED	(3) NUMBER EXTRA	(4) RATE	(5) TOTALS
TOTAL CLAIMS	20 — 20	0	$18	$ 0
INDEPENDENT CLAIMS	3 — 3	0	$78	$ 0
ANY MULTIPLE DEPENDENT CLAIMS	0		$260	$ 0
BASIC FEE: Design $310.00); Utility $690.00)				$ 690
TOTAL FILING FEE				$ 690
OTHER FEES				$
TOTAL CHARGES TO DEPOSIT ACCOUNT				$ 690

Charge $__690__ to Deposit Account 08-2025. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16, 1.17, 1.19, 1.20 and 1.21. A duplicate copy of this sheet is enclosed.

"Express Mail" label no. EL634216665US

Date of Deposit July 31, 2000

I hereby certify that this is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231.

By Laura M Clark

Typed Name: Laura M. Clark

Respectfully submitted,

Jeffrey B. THOMPSON

By Les Gehman

Leslie P. Gehman

Attorney/Agent for Applicant(s)
Reg. No. **45,624**
Date: July 31, 2000

Telephone No.: **(970) 898-3642**

Rev 12/99 (TransNew)

- Attach as First Page to Transmitted Papers -

Method And System For Extendable Class-Based Shared Data-Types

Field of Invention

The present invention relates generally to software systems. More particularly, it relates to a software method and system for handling versioning problems in object oriented programming.

Background

In the field of computer networking, hardware in a system may be interconnected. For example, a server may be connected to client terminals, which may themselves be connected directly or through the server to peripheral devices. Communication between these interconnected hardware devices is controlled through software. With respect to the transmission of data from one client to another using object-oriented programming, the software objects may be serialized and transmitted through the server to the recipient client. In order for the server to process the transmitted data and in order for the recipient client to understand the data, the system must have a uniform definition of the object.

In interconnected systems, a versioning problem exists when the hardware or the software of one device is upgraded to a new version while other devices operate using an older version. In a system having multiple interconnected devices, it is possible for several different versions of the hardware and/or software to be running at the same time. In order for these devices to effectively communicate with each other, they must share some commonalities. For example, in a system using object oriented programming that defines certain classes, every device in the system must recognize the class; otherwise, the devices cannot properly communicate with each other. In a system that defines classes, both the sender and the recipient must understand the class and must have consistent definitions of those classes so that they may be properly interpreted. A versioning problem results when a newer version of a hardware or software device uses classes that are not defined in the earlier versions.

The problem with using different versions is that a user of an older version might not be able to recognize new classes that are defined in the newer version, because they were not defined in the original software. The problem may likewise affect the server that delivers the data from the sending client to the recipient client. If the server is of an older version and does not recognize the new classes of objects

transmitted by the sending client, then the server may refuse to deliver the data to the recipient because it may lack the understanding of the data. The results of this versioning problem vary in existing systems. In some instances, the recipient client might receive the data, but will be unable to process it. In worse cases, existing clients, servers, or other devices in the system may entirely shut down if they do not recognize a new class definition of an object.

What is needed is a system for handling versioning problems that enables a user of an older version to receive new class definitions without completely shutting down. In particular, what is needed is a system that defines software objects such that new classes can be added by leveraging known definitions, such that an older version of a device will recognize and process some part of the new object classes without faulting.

Summary

A method and software system are disclosed for creating and using an extendable class-based shared data-type in object oriented programming to overcome versioning problems between interconnected devices. An extensible object class is created by defining a data-type having a base class, a type identifier, and a space reserved for additional properties. The base classes are known by all devices in an original version. New classes are added by using an existing base class and adding additional attributes to the additional properties portion of the new data-type. The type identifier is a unique number or string that identifies the data-type and certain information regarding the base class. An older version of a device that receives a new data-type from a newer device can determine the base class properties of the new data-type by reference to a catalog. Based on this base class information, the older version of a client or server can process the new data-type. Even if the older version cannot accommodate the new attributes, recognition of the base classes ensures that the device will not halt the transfer of information and that it will process the data-type to the extent that it recognizes its properties.

Summary of Drawings

Figure 1 shows a block diagram of the structure of a data-type of an object.
Figure 2 shows a block diagram of the system having a set of common base classes.
Figure 3 shows a block diagram of the system having a server and multiple clients.

Description

In object-oriented software, such as Java or C++, classes are used to define software objects. The class definitions include the data attributes of the class and also include methods of the object. An object is a particular item defined by the class. A particular object is represented as an instance of the class, having certain values. The system forms classes and structures by specifying base functionality at the programming level. This allows the addition of data elements to the base classes. A computer system implementing the data-type method allows different versions of a client and a server to share a common understanding of changing definitions, because the base class is known by both old and newer versions.

A computer system in which data-type object-oriented programming is used creates extendable class-based data-types, by defining software objects as data-types, where the data-type is a set of meta-information about a base class. Figure 1 shows the format of a data-type 10 having a base class 12, a type identifier 14, and an additional properties portion 16. The base class 12 forms the common definitions of certain base objects that are used in the system. The base class 12 defines general properties of the objects that are used by the system 50, shown in Figure 2. In one embodiment, all objects defined in an original data-type version and in later versions will use a common set of base classes 12.

The type identifier 14 provides a common means of identifying each data-type 10 so that the data-types 10 may be cataloged by a server or a client and cross-referenced. From the type identifier 14, the base class 12 or its properties can be determined. The devices, such as the client or the server, can maintain a catalog of the existing data-types organized by their type identifiers 14. Based on the type identifier 14, an older version of hardware used in the system can recognize common base characteristics of a newer data-type 10.

The additional properties portion 16 further defines the specific attributes of an object. Each additional property 16 is a name-value pair, where the name is an identifier (typically a string) and the value is an object of any class known to all clients and servers. The additional properties portion 16 also serves as a space reserved for adding modifications to the data-type 10. Each base class 12 may have a way to store, process, and set additional properties 16.

Servers 30 and clients 20 (shown in Figure 2) may have a mechanism for sharing the data-type 10 information. The servers and clients may be able to create

and accept new data-types 10. However, in one embodiment, early versions of the server and clients might not be required to allow users to create, manipulate, or use new data-types 10. In one embodiment, the system 50 may be able to handle new data-types 10 from later versions, for example by sharing only base-class 12 properties or by indicating an inability to handle the type 10.

To make new data-types 10 that can be recognized by earlier versions, data-types 10 are defined to include an additional properties portion 16 that is reserved for attributes that may define new data-types 10 added by future versions. A future client or server version may define new objects 10, or data-types 10, having new properties designed to perform a new function. A new data-type 10 builds on the base class 12 by reusing existing base classes 12 and adding additional properties thereto. The existing base class 12 forms a common threshold of knowledge between older and newer versions of the server and client because data-types 10 used by all versions of the server and client will have common base class 12 definitions. A new data-type 10 is created in newer versions by adding the new attributes to the additional properties portion 16 of the data-type 10. A new data-type 10 has a known base class 12, plus additional name-value pairs as additional properties 16.

By using a common base type 12, the system 50 avoids the problem of non-recognition of new data-types 10 by older versions of clients and servers by providing a common understanding between old and new data-types 10. When an older version of a client or server receives a newer data-type 10, the older server or client will, by definition, recognize the base class 12, even though the older version of the client or server does not recognize the new attributes contained in the additional properties portion 16 of the data-type 10. Because the older version of the client or server recognizes the base class 12, the client or server knows how to process the new data-type 10 and can perform limited functions on it.

Figure 2 shows an interconnected system 50 having a client 20 and a server 30. In this embodiment, both the client 20 and the server 30 have a data-type 10 storage and handling unit 22, 32, respectively. The handling and storage units 22, 32 are also referred to as catalogs that cross-reference type identifiers 14 with their respective base classes 12 and data-types 10. Both the client 20 and the server 30 store class objects of various data-types 10 available to the client 20 or to the server 30, respectively. The client 20 and the server 30 share a common set 40 of known

base classes 12. The client 20 is also shown to have user interface representations 24 of the objects that are exchanged. Based on the version of the client 20, the user interface representations 24 may vary, and in some older versions certain representations of newer objects or properties may not be implemented.

Figure 2 shows the interaction between a client 20 and a server 30 in the system 50. By way of example, the system 50 may be originally shipped from the factory having the base server 30 and the base client 20. In use, software objects are serialized and sent between the client 20 and server 30. In this context, the client 20 and server 30 need to know how to reconstitute the software objects. This is done using known definitions of software objects. In their original versions, both the server 30 and the client 20 will know the original set of base classes 40, and both will have a set of data-types 10 in catalogs based upon what the server 30 and the client 20 know about the base classes 40. The set of data-types 10 will correspond to the base classes 12. After a period of time, a newer version of the hardware or the software is used in the system 50. Some users of the system may determine that additional features are useful and have added these features to newer versions clients 20 or servers 30, thereby creating a versioning problem.

Figure 3 shows a system having the server 30 and multiple clients 20. A client 20' may be a newer version of the client 20 and may have new data-types 10 (see Figure 2) defined to perform new functions. In use, the new client 20' may send a new data-type 10 to an older version of the client 20 through the server 30. The new data-type 10 has certain known properties that are part of the base class 12, as well as certain additional properties 16 that are not known by the older versions of the client 20 and the server 30. For example, an older version of the client 20 might receive a new data-type 10 having the base class 12 properties A, B, and C, and having the newly added, additional attributes D and E. The client 20 will recognize the known properties A, B, and C and will process the properties A, B, and C. If, for example, all of the properties are fields that are displayed, then the older client 20 might display fields A, B, and C, and might not display new, undefined fields D and E. An older version of a server 30 might receive a new data-type 10, and would generally have the ability to process the data-type 10, according to its base class 12.

In one example, a software system may be adapted for word processing in a notes program that formats electronic notes, displaying information in a user-friendly

manner. Initially, the system may have three data-types 10 set: general notes, issues, and decisions. Each of the data-types 10 may be used to create a different type of note. Each data-type 10 is distinguished by a unique type identifier 14, and each may have different base classes 12, which may share certain properties common to "notes." In this embodiment, the additional properties portion 16 of each data-type 10 may be empty, because these data-types 10 are original and therefore known to all versions fo servers 30 and clients 20. Users of a system may send notes of any of these formats to other users through the server 30, and each server 30 or client 20 device will understand these formats, because they were included in the original base classes.

If, however, the newer client version 20' creates a new type of note, a "proposal type" note for example, then a potential versioning problem arises. When one client 20 creates an object of a known class, such as general notes, the client 20 can send the general note across the server 30 to another client 20, and the recipient client 20 can recognize and process the general note because the recipient client 20 has known class definitions. However, when a newer client version 20' creates a new object, such as the proposal, the recipient client 20 cannot properly process the proposal if the recipient has an older client version 20. If the server 30 is also of an older version, it too may have difficulty processing the proposal type in existing systems. An older version of a server 30 may refuse to deliver the unknown proposal type to another client.

The system allows older versions to extract as much recognizable data from the unknown objects and prevents the newer version from refusing to accept the unknown object. By providing an additional properties portion 16 of data-types 10, the new proposal object can be defined in terms of a known base class 12. In the example of the proposal data-type 10, this data-type 10 might have a known base class 12 having the same general properties as one of the other notes data-types 10. The new attributes that distinguish the proposal type from other data-types 10 of the notes class would be located in the additional properties portion 16 of the data-type 10. The type identifier 14 would instruct recipient clients 20 of the base class 12 characteristics of this new data-type 10. Using the type identifier 14 in conjunction with a catalog of known base types 12, the older client 20 or server 30 can recognize certain common properties or fields present in the new object. The recipient client 20 may be designed to process the new data-type 10 to the extent that it recognizes the

fields or properties of the base class 10. A server 30 may be designed to process the new data-type 10 as any other notes data-type 10.

In another embodiment, similar data-types 10 might be initially designed having certain generic base classes 12, along with additional attributes 16. New data-types 10 added in later versions would have the option of not only adding additional information to existing base classes 12, but could also take away original additional properties 16 to the extent that the new data-type 10 does not use them. In the example above, these three data-types 10 might all have the same base class 12 pertaining to notes. The additional properties portion 16 of each data-type 10 may define the particular nature of each of the original three types of notes. By including additional properties 16 in the initial data-types 10, the system creates a way for the older versions to recognize certain characteristics of the new object.

Although the present invention has been described in detail with respect to certain embodiments thereof, variations are possible. The present invention may be embodied in other specific forms without departing from the essential spirit or attributes thereof. By way of example, the system has been described particularly with respect to Java programming and to specific illustrations. One skilled in the art will recognize that the system applies to all forms of object-oriented programming and to various implementations. It is desired that the embodiments described herein be considered in all respects as illustrative, not restrictive, and that reference be made to the appended claims for determining the scope of the invention.

Claims

1. A method for creating extendable object-oriented code comprising:

defining a data-type to include a base class and an additional properties portion;

defining the base class in terms of known properties; and

reserving the additional properties portion for a future modification to the data-type.

2. The method of claim 1, wherein the base class includes properties that are capable of being understood by a plurality of interconnected devices.

3. The method of claim 2, further comprising creating a new data-type by

using an existing base class; and

adding a new attribute to the additional properties portion.

4. The method of claim 3, wherein the additional properties portion comprises a name-value pair.

5. The method of claim 2, further comprising defining the data-type to include a type identifier that provides a unique identification for a data-type.

6. The method of claim 5, wherein the base class can be determined from the type identifier by using a catalog that is maintained in each of the plurality of interconnected devices.

7. The method of claim 6, wherein the method is implemented on a computer system in a Java programming language.

8. A method for creating an extendable class-based shared data-type for use in object oriented programming comprising:

creating a first data-type comprising a first base class that includes initial properties; and

creating a second data-type comprising:

the first base class; and

a second additional properties portion having a new attribute.

9. The method of claim 8, wherein the first data-type further comprises a type identifier that provides information about the first base class.

10. The method of claim 9, wherein the information is unique to the data-type and can be used by a recipient to determine the properties of the data-type.

11. The method of claim 10, wherein the recipient can cross-reference the type identifier using a catalog to determine properties of the first base class.

12. The method of claim 8, wherein the first data-type further comprises a first additional properties portion.

13. The method of claim 8, wherein the new attribute is a name-value pair.

14. A software system comprising:

a storage medium; and

a software program stored on the storage medium for creating an extendable object-oriented data-type, wherein the data-type as comprises:

a base class that defines base characteristics of the data-type; and

an additional properties portion.

15. The software system of claim 14, wherein the data-type further comprises a type identifier.

16. The software system of claim 15, wherein the type identifier is unique to the data-type and provides information regarding the data-type.

17. The software system of claim 16, wherein the type identifier can be cross-referenced using a catalog to determine known characteristics of the data-type.

18. The software system of claim 17, wherein the additional properties portion is capable of holding a new attribute having a name-value pair.

19. The software system of claim 18, wherein the base class is known to a plurality of interconnected devices on which the software system is used.

20. The software system of claim 19, wherein the new attribute may not be known to at least one of the plurality of interconnected devices.

Abstract

A method is disclosed for creating and using an extendable class-based shared data-type in object oriented programming to overcome versioning problems between interconnected devices. An extensible object class is created by defining a data-type having a base class, a type identifier, and a space reserved for additional properties. The base classes are known by all devices in the original version. New classes are added by using an existing base class and adding additional attributes to the additional properties portion of the new data-type. The type identifier is a unique number or string that identifies the data-type and certain information regarding the base class. An older version of a device that receives a new data-type from a newer device can determine the base class properties of the new data-type by reference to a catalog. Based on this base class information, the older version of a client or server can process the new data-type. Even if the older version cannot accommodate the new attributes, recognition of the base classes ensures that the device will not halt the transfer of information and that it will process the data-type to the extent that it recognizes its properties.

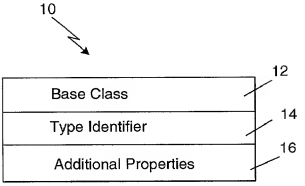
10
12
Base Class
14
Type Identifier
16
Additional Properties

Fig. 1

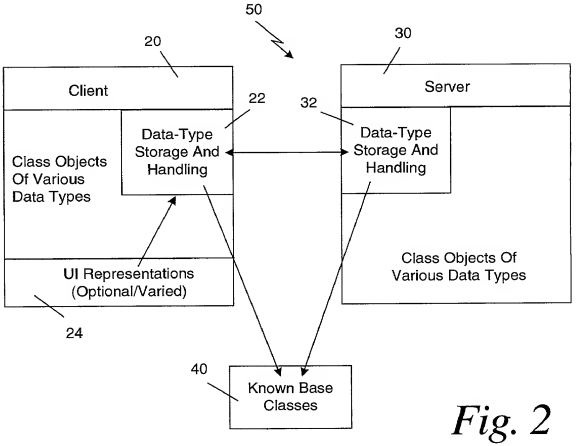
50
20
30
Client
22
32
Server
Data-Type Storage And Handling
Data-Type Storage And Handling
Class Objects Of Various Data Types
UI Representations (Optional/Varied)
Class Objects Of Various Data Types
24
40
Known Base Classes

Fig. 2

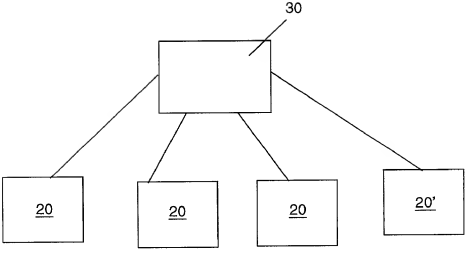

Fig. 3

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

ATTORNEY DOCKET NO. 10003696-1

As a below named inventor, I hereby declare that:

My residence/post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

Method And System For Extendable Class-Based Shared Data-Types

the specification of which is attached hereto unless the following box is checked:

( ) was filed on _____ as US Application Serial No. or PCT International Application Number _____ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understood the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose all information which is material to patentability as defined in 37 CFR 1.56.

Foreign Application(s) and/or Claim of Foreign Priority

I hereby claim foreign priority benefits under Title 35, United States Code Section 119 of any foreign application(s) for patent or inventor(s) certificate listed below and have also identified below any foreign application for patent or inventor(s) certificate having a filing date before that of the application on which priority is claimed:

COUNTRY	APPLICATION NUMBER	DATE FILED	PRIORITY CLAIMED UNDER 35 U.S.C. 119
N/A			YES ___ NO: ___
			YES ___ NO: ___

Provisional Application

I hereby claim the benefit under Title 35, United States Code Section 119(e) of any United States provisional application(s) listed below:

APPLICATION SERIAL NUMBER	FILING DATE
N/A	

U. S. Priority Claim

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

APPLICATION SERIAL NUMBER	FILING DATE	STATUS (patented/pending/abandoned)
N/A		

POWER OF ATTORNEY:

As a named inventor, I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith:

Customer Number 022879

Place Customer Number Bar Code Label here

Send Correspondence to:
HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80528-9599

Direct Telephone Calls To:
Leslie P Gehman
(970) 898-3642

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Inventor: Jeffrey B. Thompson **Citizenship:** US

Residence: 4348 Westbrooke Court Fort Collins, CO 80526

Post Office Address: Same as residence

Inventor's Signature **Date**